Available online at www.sciencedirect.com
SCIENCE DIRECT®
Future Generation Computer Systems 21 (2005) 115–123
FGCS FUTURE GENERATION COMPUTER SYSTEMS
www.elsevier.com/locate/future

File-based replica management

Peter Kunszt[a,*], Erwin Laure[a], Heinz Stockinger[a], Kurt Stockinger[b,1]

[a] *CERN, European Organization for Nuclear Research, CH-1211 Geneva 23, Switzerland*
[b] *Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, USA*

Abstract

Data replication is one of the best known strategies to achieve high levels of availability and fault tolerance, as well as minimal access times for large, distributed user communities using a world-wide Data Grid. In certain scientific application domains, the data volume can reach the order of several petabytes; in these domains, data replication and access optimization play an important role in the manageability and usability of the Grid.

In this paper, we present the design and implementation of a replica management Grid middleware that was developed within the EDG project [European DataGrid Project (EDG), http://www.eu-egee.org] and is designed to be extensible so that user communities can adjust its detailed behavior according to their QoS requirements.

Keywords: Data Grid; Replica management; Replica access optimization

1. Introduction

Grid computing addresses the issue of distributed computing over the wide-area network that involves large-scale resource sharing among collaborations of individuals or institutions. Data Grids, in particular, address the needs of data intensive applications that deal with the evaluation and mining of large amounts of data in the terabyte and petabyte range.

One of the principal goals of Data Grids is to provide easy-to-use, transparent access to globally distributed data and to abstract the complexities from the user and the applications. Essentially, it aims at making data access and location as easy as on a single computer.

Optimization of data access can be achieved via data replication, whereby identical copies of data are generated and stored at various sites. This can significantly reduce data access latencies. However, dealing with replicas of files adds a number of problems not present when only a single file instance exists. Replicas must be kept consistent and up-to-date, their locations must be traced, their lifetimes need to be managed, etc.

Replica management is achieved by coordinating underlying Grid services like file transfer services and protocols (GridFTP) [3], replica and metadata catalogs [8]. These services have to be properly secured (GSI).

* Corresponding author.
E-mail address: peter.kunszt@cern.ch (Peter Kunszt).
[1] The work was done when Kurt was affiliated with CERN.

0167-739X/$ – see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2004.09.017

Achieving replica management by serializing these services is error prone and tedious. There is a clear need to automate the process, providing a service that can be driven by higher level policies, easily understood by the end-user. As an example, consider the task of replicating a file. This requires the application to perform a wide area transfer (e.g. using GridFTP) and to update various catalogs while checking access rights and managing errors and failures. Another example is replica selection: without a higher level Grid service to call on, the users will need to implement their own replica selection algorithms to locate appropriate data, increasing the burden on the application programmers.

In this paper, we present the design and implementation details of a replica management service (RMS, originally presented and introduced in [15] and [17]) that is intended to provide the application programmer as well as an end-user with an easy-to-use, intuitive interface, hiding the details of the underlying services. We have designed and implemented a prototype in the context of the EU DataGrid project (EDG) [12]. The remainder of this paper is organized as follows: we present the design of our replica management framework in Section 2 and its latest implementation in Section 3. Results obtained with this prototype are reported in Section 4. After a discussion of related work (Section 5), we end this paper with a brief summary and an outlook on future work.

2. Design of a replica management framework

In [15,17], we presented the design of a general replica management framework. In that model, a replica management service contains all the logic to coordinate the underlying services, providing the user with a unified interface to all replica management functionalities. Fig. 1 presents the user's perspective of the logical layout of the components of our replica management system. The entry point to all services is the *Replica Management Service* which interacts with the subcomponents of the system.

- The *Core* module coordinates the main functionality of replica management, which is replica creation, deletion, and cataloging by interacting with third party modules. These external modules include transport services, replica location services (RLS), meta-data services for storing replication meta-data such as file meta-data (size, checksum, etc.), management meta-data, and security meta-data (such as access control lists), and processing services that allow pre- and post-processing of files being replicated.
- The goal of the *Optimization* component is to minimize file access and/or transfer times by pointing access requests to appropriate replicas and pro-actively replicating frequently used files based on access statistics gathered.
- The *Consistency* module takes care of keeping the set of replicas of a file consistent as well as the meta information stored in various catalogs.
- The *Subscription* module takes care of subscription-based replication where data appearing at a data source are automatically replicated to subscribed sites.
- The *Session Management* component provides generic check-pointing, restart and rollback mechanisms to add fault tolerance to the system.
- *Collections* are defined as sets of logical filenames and other collections. Collections are particularly important if end-users need a higher level data access granularity than just single files.
- The *Security* module manages the required user authentication and authorization, in particular, issues pertaining to whether a user is allowed to create, delete, read and write a file.

The model of providing high-level functionality by coordinating a set of underlying services is one of the paradigms of Grid computing and other service-oriented architectures. By following this paradigm, we allow for deployments providing customized levels of Quality of Service as we foresee that Virtual Organizations will use the Replica Management Services in very different ways. Only the core, optimization and security modules have been actually implemented and deployed in the EU DataGrid.

3. Implementation details

We implemented the replica management system within the framework of the EU DataGrid project. Starting from the preliminary architecture presented in [15], we implemented a first prototype that used the

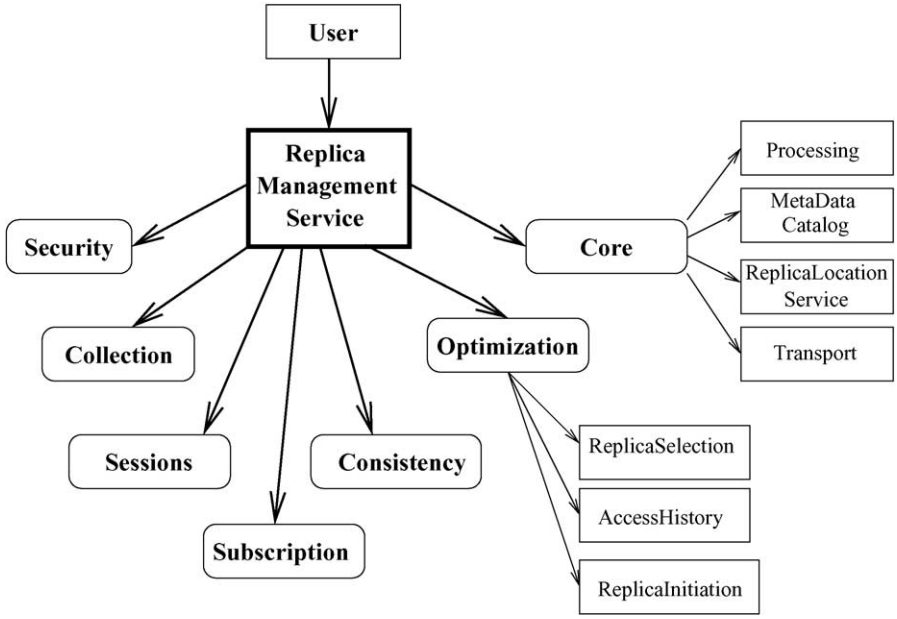

Fig. 1. Main components.

code name “Reptor”. Preliminary implementation details and results have then been presented in [17]. During the final phases of the EDG project, feedback from several user communities has been collected and the implementation has evolved as outlined in [7]. In the following, we give more implementation details that correspond to the final version of the code developed within the EDG project.

Most components and services depicted in Fig. 1 are implemented in the Java language. Services such as the Replica Location Service, the Replica Optimization Service (ROS), etc. are deployed as web services. We initially depended only on open source technologies like the Jakarta Tomcat servlet container, the MySQL database and Apache Axis. However, for high performance and availability reasons, some of the web services have later been ported to the Oracle Application Server as well as the Oracle 9i database.

The current implementation provides the core module (interfacing to the transport and catalog services) and the optimization component, leaving the collection, session management, subscription and consistency modules for future versions. One particular change of the implementation with respect to the original design is that the logic of the coordination replica management service is implemented on the client-side rather than offering a separate service; this is to defer the issues we originally faced with the details of authorization with respect to delegation of rights to services as described in [7].

Security is provided within the Grid Security Infrastructure (GSI) framework. Our web services are able to authenticate Grid certificates via a trust manager, an authentication module which can be plugged into web service containers. We also have a fine grained authorization mechanism in place that can interpret certificate extensions provided by the Virtual Organization Membership Service, VOMS [2]. The advantage of such an approach is that our security infrastructure is backward compatible with the existing GSI implementations that simply ignore these extensions.

In wide-area distributed computing, web service technologies are becoming increasingly popular, since they provide easy and standardized access to distributed services in a multi-lingual, multi-domain environment. The same is true in Grid computing where the upcoming OGSA standard [16] aims at leveraging web services in the Grid context. Due to the general recognition of the web service standard and to be prepared to move

to the OGSA standard, we have chosen to adopt the web services paradigm.

3.1. Interaction with external services

As described above, Reptor needs to interact with many Grid services such as the replica location service and the information services. We have implemented Reptor as set of modules that is easy to extend and interface to other Grid components.

We have defined interfaces for each Grid service that Reptor is accessing. In the spirit of general software engineering principles as well as OGSA, we have made all these modules pluggable: if one of the services needs to be replaced by another service providing the same functionality, the appropriate implementation of the interface needs to be provided to Reptor, which can be configured to load the given implementation instead of the default ones. Currently, Reptor has been tested and used with the following Grid services:

- *Replica Location Service* [8] as the replica catalog service: used for locating replicas by their unique identifiers (GUIDs);
- *Replica Metadata Catalog* (*RMC*) stores the user-definable logical file name aliases to GUIDs and simple meta data on replicas, e.g. the owner, file size and time stamps;
- *Information Service*: used for obtaining information about the topology of the Grid. In EDG project, two different implementations have been used and interfaced to:
  - EDG's *Relational Grid Monitoring Architecture* (*R-GMA*) [11],
  - Globus' Monitoring and Discovery Service (MDS) [14];
- The *EDG Network Monitoring Services* [6] providing statistics about network characteristics;
- The *EDG Storage Element* (*SE*) *Services* [6] providing information about the storage latency. In detail, the Storage Element service can also have a Storage Resource Management (SRM) [20] interface to a Mass Storage System.

Some of these services are currently client/server architectures using proprietary protocols for communication, but most of them will in the near future turn into proper web services, advertising their interface through WSDL.

3.2. The Reptor client

We provide a command line interface and a Java API; C and C++ APIs are also available for parts of the interfaces, and it is straightforward to generate bindings for other languages thanks to the web service technologies used. The client is a thin layer that interacts with the services mentioned above, acting as the coordinator of the replica management subservices.

Originally, we had a pure Java version of the client service using the Commodity Grid for Java *CoG* [26] that provided GridFTP client functionality. However, in order to be able to support parallel streams for better performance, we can also configure the Reptor client to use the native GridFTP C libraries where available.

Currently, the client provides the following functionalities:

- *Copy* a file between two sites using third-party transfer [3] if required.
- *Register* an existing file in the replica location service to allow its retrieval by other Grid services. A GUID is assigned to the file and registered in the meta-data catalog. It is used as the immutable unique identifier of the file and its replicas. Since GUIDs are typically not human-readable, the user may provide a logical identifier on its own which acts as an alias of the GUID. It must be unique as well, but unlike the GUID, it can change in time.
- *Create replicas* of a file that has previously been registered in the replica location service. The optimization service discussed below can be exploited for selecting an existing replica to be copied to the destination location.
- *Delete* a replica which results in the physical deletion of the file and its removal from the associated catalogs.
- *Unregister* a file from the catalog without physically deleting the file.
- *List* all the replicas that have been registered by a given identifier (GUID), or look up the logical names or GUID based on a replica.
- *Select* the "best" replica to be used on a given site with the help of the optimization service (see below).

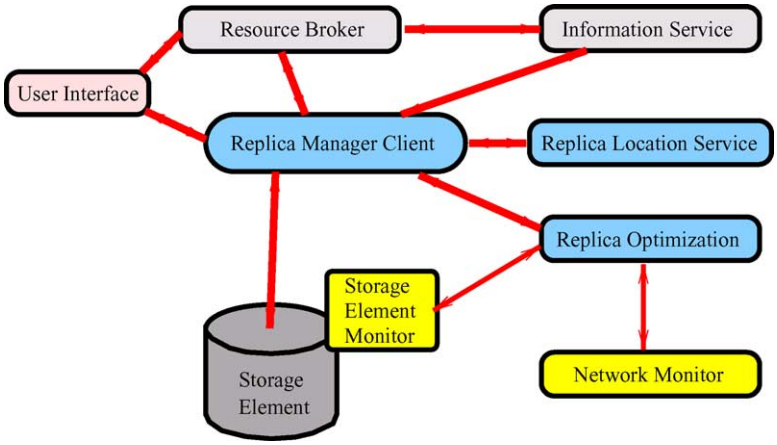

Fig. 2. Interaction of Replica Manager with other Grid Components.

- Retrieve the *access costs* for all available replicas of a given file with respect to specified sites.

3.3. Replica Optimization Service

The goal of the *Replica Optimization Service* is to select the best replica with respect to network and storage access latencies. In other words, if for a given file, several replicas exist, the optimization service determines the replica that should be accessed from a given location. Similarly, the optimization service might also be used to determine the best location for new replicas. We currently do not take into account an eventual possibility to access the data directly over the wide area but assume that in order to access a file, it has to be available in the local area network.

The Replica Optimization Service is implemented as a light-weight web service (called *Optor*). It gathers information from the EDG network monitoring service and the EDG storage element monitoring service about their network and storage access latencies. Based on this information, Optor takes a decision which network link should be used in order to minimize the transfer time between two end points as described in [6].

Apart from selecting replicas and storage locations, Optor also provides methods to retrieve estimated file access costs. These can be exploited by other Grid services, such as meta schedulers like the EDG Resource Broker [10]. Based on the information obtained by Optor, the broker can schedule jobs to sites that allow efficient file access while maximizing the overall throughput. Thus, the Replica Manager, in particular its optimization component, assists the Resource Broker in the job scheduling process.

The interaction of the Grid components in the overall EDG project architecture is shown in Fig. 2. Note that the Storage Element Monitor (a service that returns access latencies for files stored in mass storage systems) has only been partly interfaced with an Storage Access Optimization Service. We have reported earlier work on that in [24].

4. Experimental results

In this section, we report on the results of tests carried out with Reptor to perform *optimized replication* of files. In particular, we discuss the performance of the core functionalities and the optimization component. For detailed performance figures on the interaction with the Replica Location Service as well as the Replica Metadata Catalog, we refer the reader to [7].

4.1. Testbed setup

All our experiments were run on five major sites of the EU DataGrid testbed in France, Italy, the Netherlands, Switzerland and the UK (see Fig. 3: note that there are many more sites in different countries and locations but we restrict the experiments only to the five major sites). Each site consisted of *Computing Elements* (CE) and *Storage Elements*. A Computing Ele-

Fig. 3. Major testbed sites of the EU DataGrid project.

ment typically runs a (Globus) gate-keeper and a local scheduling system like PBS or Maui. A Storage Element is a generic Grid interface to storage, i.e. it can be a single disk, a disk farm or a mass storage system. A typical Grid job is submitted from a *User Interface* to the Resource Broker and is scheduled to be run at a Computing Element based on information provided by the Information Services and the Replica Manager (see Fig. 2).

The components needed by the Replica Manager Client were deployed at dedicated hardware at CERN, each machine running a single service only. We had one machine each to run the User Interface (i.e. the Replica Manager Client), the Replica Location Service, the Replica Metadata Catalog, the Replica Optimization Service and the Information Service (R-GMA in our case [11]).

The network monitoring components have been deployed at each testbed site by the EDG networking work package. We make use of their network monitoring information provider which is accessible at IN2P3 in Lyon. The network monitors keep track of the performance of the network over time which is usually changes significantly over time—this motivates our strategy of optimization based on recent network metrics.

4.2. File replication

In this section, we provide performance numbers for replicating files of various sizes over a wide-area network including updating the replica catalog. We compare these numbers with raw data transfers achieved via GridFTP. Note that Reptor adds some overhead to the basic file transfer due to various consistency checks: Reptor checks whether the file already exists on the destination before starting the copy, obtains file meta-data and checks the file existence after having performed the copy. This overhead is constant, independent of the actual file size.

Reptor provides two different replication options, *conventional replication* and *optimized replication*. The conventional option takes source and destination SEs as input parameters. The optimized option takes a logical file name (LFN) and the destination SE as input parameters and chooses the best replica as described above: for a given LFN, all physical replicas are identified through the RLS and for each replica, the transfer costs to the specified destination SE are calculated using the Network Monitor. The replica having the minimal cost is used as the source file for replication.

By having both options, users may either control the source and destination of the copy themselves or let the system decide on the best resources to use on their behalf.

In Fig. 4, we show our measurements for replicating files between three sites within the EU DataGrid testbed. The file sizes range from 100 MB to 1 GB. We use GridFTP as the file transfer mechanism with eight parallel streams. Note that Reptor adds some overhead to the basic file transfer due to various consistency checks: Reptor checks whether the file already exists on the destination before starting the copy, obtains file meta-data and checks the file existence after having performed the copy. This overhead is constant, independent of the actual file size. For replicating 1 GB files, we can observe an overhead of around 10% due to the additional consistency checks. If needed, the file checksumming can also be switched off in order to reduce the consistency overhead.

The next experiment shows the possible performance gain one can obtain by optimized replication (see Fig. 4d). Due to the different network bandwidths within the testbed and their rapid change in time, we gain factors of two and more in transfer efficiency by automatically choosing the best network link for file replication. If the users have to specify the source for the copy by themselves, they can achieve at best the same result. Without an automatic replica selection mechanism, they most likely will choose a suboptimal

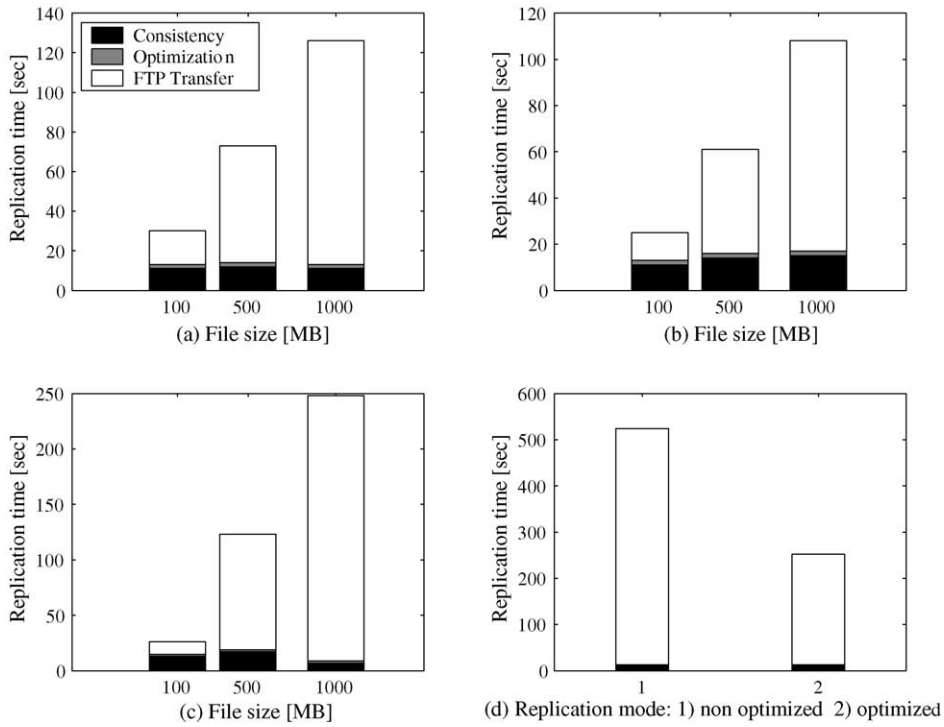

Fig. 4. Performance measurements of replicating files with sizes 100 MB, 500 MB and 1 GB between (a) CERN and RAL, (b) RAL and NIKHEF, and (c) NIKHEF and CNAF. (d) Performance measurements of replicating 1 GB files to NIKHEF (1) using conventional replication: IN2P3 was specified as the source and (2) using optimized replication: RAL was picked as the better source automatically.

network link. In both cases, the consistency overhead can be neglected, since it is identical for both replication scenarios.

To sum up, we demonstrated network optimized replication with Reptor and achieved a substantial gain in efficiency, especially in a distributed Grid environment where the network latencies are subject to rapid change. We have found that the changes are slow enough so that recent network monitoring information is usually still applicable at the time of the file transfer.

5. Related work

We have performed early prototype work with a data replication tool named *Grid Data Mirroring Package* (*GDMP*) [23]. It is a basic file replication tool with support for mass storage system staging and automatic replication through a publish–subscribe notification system. Another predecessor of Reptor is the edg-replica-manager [22], simple client replication tool.

The Globus Alliance provides a Reliable File Transfer (RFT) service [19] that is part of Globus Toolkit 3. Replica management functionalities are supposed to be built on top of RFT and therefore, would fit into our replica management framework. In addition, Globus implemented the Copy And Registration Service (CAR) [14], a tool that provides an OGSI-compliant mechanism for accessing existing RLSs.

An integrated approach for data and meta-data management is provided in the Storage Resource Broker (SRB) [5]. Data cataloging and access to relational and object-oriented databases are provided.

The most related work with respect to replica access optimization can be found in the Earth Science Grid (ESG) [4] project where preliminary replica selection has been demonstrated using the Network Weather Service (NWS) [27]. Within the Grid and High-Energy Physics community, the most closely related projects are Sequential data Access via Metadata (SAM) [25], Alien [1] and gLite [13]. SAM was initially designed to handle data management issues of the D0 experiment

at Fermilab. It also addresses replication job submission issues. Both Alien and gLite provide replication services and some data access optimization.

Since file replication often deals with large storage systems, local storage resource management becomes vital and we participate in the standardization effort of the Storage Resource Management [20] specification that builds the natural link between Globus, Grid-wide replica management and local file management. Similar to the SRM standardization, a new standardization effort has started for Replica Management Services [21] with the goal of integrating existing data management software based on standard interfaces.

6. Conclusion and future work

We have presented the design and implementation of a high level replica management tool providing the functionality required for efficient data access in a Grid environment. We have discussed in detail our optimized replica selection mechanism that takes into account network monitoring data in order to identify the optimal network link with respect to data transfer. The experimental results show that our approach significantly reduces wide area transfer times.

The framework is easily extensible and allows for many different deployment scenarios to serve the needs of many different Virtual Organizations.

Currently, we have implemented the core and optimization functionality of the replica management design. The replication system has been successfully deployed and used in the EDG project and therefore by many users in the High-Energy Physics, Bioinformatics and Earth Observation domains. Since the EDG project has formally finished, the software is now further evolved and enhanced by follow-up projects such as LCG [18] and EGEE [9]. Both of them have the aim to provide a large-scale Grid infrastructure for data intensive sciences.

Acknowledgments

We would like to thank first and foremost our colleagues in WP2 who have had contributions to the design and development of Reptor and for running the experimental testbed. We would like to thank our colleagues in the EU DataGrid project who provided us with invaluable feedback on our existing replica management tools. We are also thankful to our colleagues from the Globus Alliance for many stimulating discussions. This work was partially funded by the European Commission program IST-2000-25182 through the EU DataGrid Project.

References

[1] AliEn Project, http://alien.cern.ch.
[2] R. Alifieri, et al., An authorization system for virtual organizations, in: European Across Grids Conference, in: Santiago de Compostela, Spain, February, 2003.
[3] B. Allcock, et al., Efficient data transport and replica management for high-performance data-intensive computing, in: Proceedings of the 18th IEEE Symposium on Mass Storage Systems and 9th NASA Goddard Conference on Mass Storage Systems and Technologies, San Diego, April 17–20, 2001.
[4] B. Allcock, et al., High-performance remote access to climate simulation data: a challenge problem for data grid technologies. in: Supercomputing 2001, Denver, Texas, November, 2001.
[5] C. Baru, R. Moore, A. Rajasekar, M. Wan, The SDSC storage research broker, in: CASCON'98, Toronto, Canada, 30 November–3 December, 1998.
[6] W.H. Bell, et al., Design of a Replica Optimisation Framework, Technical report DataGrid-02-TED-021215, CERN, Geneva, Switzerland, December 2002, EU DataGrid Project.
[7] D. Cameron, et al. Replica management in the European DataGrid Project, Int. J. Grid Comput., in press.
[8] A. Chervenak, et al., Giggle: a framework for constructing scalable replica location services, in: SC'2002, Baltimore, USA, November, 2002.
[9] EGEE: enabling Grids for E-science in Europe, http://www.eu-egee.org.
[10] EU DataGrid WP1. Definition of architecture, technical plan and evaluation criteria for scheduling, resource management, security and job description, EU DataGrid Project, deliverable D1.2, September 2001.
[11] EU DataGrid WP3. R-GMA: relation grid monitoring architecture, http://hepunx.rl.ac.uk/edg/wp3/.
[12] European DataGrid Project (EDG), http://www.eu-datagrid.org.
[13] gLite project page, http://cern.ch/glite.
[14] Globus Alliance, http://www.globus.org.
[15] L. Guy, P. Kunszt, E. Laure, H. Stockinger, K. Stockinger, Replica management in data grids, Technical report, GGF5 Working Draft, July, 2002.
[16] J. Nick, I. Foster, C. Kesselman, S. Tuecke, The physiology of the grid: an open grid services architecture for distributed systems integration. Open grid service infrastructure WG, Global Grid Forum, June 22, 2002.

[17] P. Kunszt, E. Laure, H. Stockinger, K. Stockinger, Advanced replica management with Reptor, in: Proceedings of the International Conference on Parallel Processing and Applied Mathematics, Czestochowa, Poland, September 7–10, 2003.
[18] LHC Computing Grid Project (LCG), http://cern.ch/lcg.
[19] R. Madduri, C. Hood, W. Allcock, Reliable file transfers in grid environments, in: Proceedings of the 27th IEEE Conference on Local Computer Networks, Tampa, Florida, November 6–8, 2002.
[20] A. Shoshani, et al., Storage resource management: concepts, functionality, and interface specification, in: Future of Grid Data Environments: a Global Grid Forum (GGF) Data Area Workshop, Berlin, Germany, March, 2004.
[21] A. Shoshani, A. Sim, K. Stockinger, Replica Management Component Services—functional interface specification, Technical Report LBNL-56333, Berkeley Lab, Berkeley.
[22] H. Stockinger, et al., Grid data management in action: experience in running and supporting data management services in the EU DataGrid Project, in: Computing in High Energy Physics (CHEP 2003), La Jolla, California, March 24–28, 2003.
[23] H. Stockinger, A. Samar, S. Muzaffar, F. Donno, Grid data mirroring package (gdmp), J. Sci. Programming 10 (2) (2002).
[24] K. Stockinger, et al., Access cost estimation for unified grid storage systems, in: Proceedings of the Fourth International Workshop on Grid Computing (Grid2003), Phoenix, Arizona, November 17, 2003.
[25] I. Terekhov, et al., Distributed data access and resource management in the DO SAM system, in: Proceedings of the 10th EEE Symposium on High Performance and Distributed Computing (HPDC-10), San Francisco, California, August 7–9, 2001.
[26] G. von Laszewski, I. Foster, J. Gawor, P. Lane, A Java Commodity Grid Kit, Gregor von Laszewski, Concurrency Comput.: Pract. Experience 13 (8–9) (2001).
[27] R. Wolski, N. Spring, J. Hayes, The Network Weather Service: a distributed resource performance forecasting service for metacomputing, J. Future Generation Comput. Syst. 15 (5–6) (1999) 757–768.

Peter Kunszt has a PhD in Theoretical Physics from the University of Bern (Switzerland). After having developed several database applications for the University of Bern's astronomy department, he moved to the Johns Hopkins University in Baltimore (USA). There, he was one of the builders of the Sloan Digital Sky Survey's Science Archive database, which has been used by all of the SDSS collaboration for several years for scientific discovery. After having moved back to CERN in Geneva (Switzerland), he started to work in the data management work package of the EU DataGrid project. He has led the work package for the second and third year of the project. Currently, he heads up the data management cluster of the EU EGEE project's middleware activity.

Erwin Laure is the deputy Middleware Manager of the EU funded project "Enabling Grids for E-Science in Europe (EGEE)" working at the European Organization for Nuclear Research (CERN). After joining CERN in 2002, he worked on data management issues within the EU DataGrid (EDG) project and later became the Technical Coordinator of EDG. He holds a PhD in Business Administration and Computer Science from the University of Vienna, Austria. His research interests include grid computing with a focus on data management in grid environments as well as programming environments, languages, compilers and runtime systems for parallel and distributed computing.

Heinz Stockinger has been working in Grid projects in Europe (European Organization for Nuclear Research, CERN) and in the USA (Stanford Linear Accelerator, SLAC) for more than 4 years. Within the EU DataGrid project (EDG), he was the Education and Outreach Manager as well as responsible for replication software in the Data Management workpackage. His research interests included distributed systems, Data Grids and in particular data replication. He is currently also external lecturer in the University of Vienna as well as the Grid Track Co-ordinator of the CERN School of Computing. Heinz holds a PhD degree in Computer Science and Business Administration from the University of Vienna, Austria.

Kurt Stockinger is a computer scientist with the Scientific Data Management Group of Berkeley Lab, Berkeley, California, where he works on database access optimization and replica management for scientific data. Typical application domains are complex astronomy, high-energy physics and combustion studies. Previously, Kurt was leading the Optimization Task of the EU DataGrid Project managed by CERN. His research interests include performance evaluation of parallel and distributed systems (Data Grids), database access optimization and multi-dimensional data structures for large data warehouses. He studied computer science and business administration at the University of Vienna, Austria, and the Royal Holloway College, University of London, England. He received a PhD in computer science and business administration from University of Vienna, Austria, under supervision of CERN's Database Group.